LISTING OF CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method of managing a relational database comprising:

a. receiving queries in a query language, the queries comprising a plurality of query terms;

b. interpreting the queries by associating at least one declarative language function with the query terms;

c. converting the queries represented by the at least one declarative language function to a plurality of imperative language statements; and

d. executing the imperative language statements.

2. (Previously Presented) The method of claim 1 comprising converting the query language to an intermediate tree representation corresponding to the at least one declarative language function associated with the plurality of query terms, and thereafter converting the query to at least one data structure that is interpreted by an imperative language interpreter core to perform the queries.

3. (Previously Presented) The method of claim 2, wherein the declarative language function is identified by a pointer to further code such that the declarative language function is treated as data within the plurality of imperative language statements.

4. (Previously Presented) The method of claim 1 wherein the declarative language function is implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

5. (Previously Presented) The method of claim 1 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

6. (Previously Presented) A method of managing a relational database comprising:

a. receiving queries in a query language;

b. converting the query language to a plurality of imperative language statements that represent a declarative language representation of the queries; and

c. executing the imperative language statements.

7. (Previously Presented) The method of claim 6 wherein the declarative language representation is chosen from the group consisting of ML, LISP, and HASKELL.

8. (Previously Presented) The method of claim 6 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

9. (Previously Presented) A method of managing a relational database comprising:

a. receiving queries in a query language;

b. representing the queries in accordance with a declarative language paradigm as a plurality of declarative language functions;

c. converting the queries represented in the declarative language paradigm to a data structure that is represented by imperative language statements; and

d. executing the imperative language statements.

10. (Previously Presented) The method of claim 9 further comprising converting the query language to an intermediate declarative representation, and thereafter converting the query to a data structure representing the intermediate declarative representation in the imperative language statements.

11. (Previously Presented) The method of claim 10, wherein the declarative language functions are identified by pointers to further code such that the declarative language functions are treated as data within the plurality of imperative language statements.

12. (Previously Presented) The method of claim 9 wherein the declarative language functions are implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

13. (Previously Presented) The method of claim 9 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

14. (Previously Presented) A database management system adapted to process queries in a pervasive computing environment, said pervasive computing environment comprising at least one client adapted to interact with a server over connection services, said at least one client controlled and configured to

a. receive the queries in a query language, the queries comprising a plurality of query terms;

b. interpret the queries by associating at least one declarative language function with the query terms;

c. convert the queries represented by the at least one declarative language function to a plurality of imperative language statements; and

d. execute the imperative language statements.

15. (Previously Presented) The system of claim 14 comprising converting the query language to an intermediate tree representation corresponding to the declarative language function, and thereafter converting the query to at least one data structure that is interpreted by an imperative language interpreter core to perform the queries.

16. (Previously Presented) The system of claim 15, wherein the declarative language function is identified by a pointer to further code such that the declarative language function is treated as data within the plurality of imperative language statements.

17. (Previously Presented) The system of claim 14 wherein the declarative language function is implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

18. (Previously Presented) The system of claim 14 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

19. (Previously Presented) A database management system adapted to process queries in a pervasive computing environment, said pervasive computing environment comprising at least one client adapted to interact with a server over connection services, said at least one client controlled and configured to

a. receive queries in a query language;

b. represent the queries in accordance with a declarative language paradigm as a plurality of declarative language functions;

c. convert the queries represented in the declarative language paradigm to a data structure that is represented by imperative language statements; and

d. executing the imperative language statements.

20. (Previously Presented) The system of claim 19 further comprising converting the query language to an intermediate declarative representation, and thereafter converting the query to a data structure representing the intermediate declarative representation in the imperative language statements.

21. (Previously Presented) The system of claim 20, wherein the declarative language functions are identified by pointers to further code such that the declarative language functions are treated as data within the plurality of imperative language statements.

22. (Previously Presented) The system of claim 19 wherein the declarative language functions are implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

23. (Original) The system of claim 19 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

24. (Previously Presented) A program product comprising computer readable program code on one or more media, said program code being capable of controlling and configuring a computer system having one or more computers to perform the process of

a. receiving queries in a query language, the queries comprising a plurality of query terms;

b. interpreting the queries by associating at least one declarative language function with the query terms;

c. converting the queries represented by the at least one declarative language function to a plurality of imperative language statements; and

d. executing the imperative language statements.

25. (Previously Presented) The program product of claim 24 comprising converting the query language to an intermediate tree representation corresponding to the at least one declarative language function associated with the plurality of query terms, and thereafter converting the query to at least one data structure that is interpreted by an imperative language interpreter core to perform the queries.

26. (Previously Presented) The program product of claim 25, wherein the declarative language function is identified by a pointer to further code such that the declarative language function is treated as data within the plurality of imperative language statements.

27. (Previously Presented) The program product of claim 24 wherein the declarative language function is implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

28. (Previously Presented) The program product of claim 24 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.

29. (Previously Presented) A program product comprising computer readable program code on one or more media, said program code being capable of controlling and configuring a computer system having one or more computers to perform the process of

a. receiving queries in a query language;

b. representing the queries in accordance with a declarative language paradigm as a plurality of declarative language functions;

c. converting the queries represented in the declarative language paradigm to a data structure that is represented by imperative language statements; and

d. executing the imperative language statements.

30. (Previously Presented) The program product of claim 29 further comprising converting the query language to an intermediate declarative representation, and thereafter converting the query to a data structure representing the intermediate declarative representation in the imperative language statements.

31. (Previously Presented) The program product of claim 30, wherein the declarative language functions are identified by pointers to further code such that the declarative language functions are treated as data within the plurality of imperative language statements.

32. (Previously Presented) The program product of claim 29 wherein the declarative language functions are implemented in a declarative language that is chosen from the group consisting of ML, LISP, and HASKELL.

33. (Previously Presented) The program product of claim 29 wherein the imperative language statements are implemented in an imperative language that is chosen from the group consisting of C, C++, Java, Modula2, and SmallTalk.